APPENDIX II

```
function  addlink(TOPO)
% addlink(TOPO)
%
%   interactively add links to the TOPO

update(TOPO);
c_src = 1;
c_dst = 2;
c_bw = 3;

figure(TOPO.cur_fig)

while (1)

fprintf(1,'\n\nHit Button 3 to end...\n\n');

% find coords and index i of src
[x1i y1i button] = ginput(1);
if (button == 3) break; end

d = sqrt((TOPO.locs(:,1) - x1i).^2 + (TOPO.locs(:,2) - y1i).^2);
[d,i] = min(d);
x1 = TOPO.locs(i,1); y1 = TOPO.locs(i,2);

% find coords and index j of dst
[x2i y2i] = ginput(1);
d = sqrt((TOPO.locs(:,1) - x2i).^2 + (TOPO.locs(:,2) - y2i).^2);
[d,j] = min(d);
x2 = TOPO.locs(j,1); y2 = TOPO.locs(j,2);

hold on;
lh = line([x1 x2],[y1 y2],'color','red');

cap = input('Enter capacity (in Mbps) > ');

fprintf(1,'About to create symetric %d Mbps link from node %d to node %d\n',cap,i,j);

doit = input('Enter Y to confirm, N to reject, and B to change bandwidth (Y)> ','s');

if (isempty(doit)) doit = 'Y'; end

if (doit == 'n' | doit == 'N')
        delete(lh);
        return;
```

```
end

if (doit == 'b' | doit == 'B')
        buf = sprintf('Enter capacity from %d to %d (in Mbps) > ',i,j);
        cap_i_to_j = input(buf);

        buf = sprintf('Enter capacity from %d to %d (in Mbps) > ',j,i);
        cap_j_to_i = input(buf);
else
        cap_i_to_j = cap;
        cap_j_to_i = cap;
end

%build the link records
clear linkab linkba;

linkab.src = i;
linkab.dst = j;
linkab.bw = cap_i_to_j;
linkab.handle = lh;

linkba.src = j;
linkba.dst = i;
linkba.bw = cap_j_to_i;
linkba.handle = lh;


% now draw the actual link on the map
delete(lh);
lh = drawlink(TOPO, linkab);

% now store the link info
TOPO.links = [TOPO.links ; linkab ; linkba ];
TOPO.linkarray = [TOPO.linkarray ; [ i j cap_i_to_j] ; [ j i cap_j_to_i ]];

end % of while loop

assignin('caller',inputname(1),TOPO);




function [C, portmap] = capacity(TOPO)
% [C, portmap] = capacity(TOPO)
% portmap maps indices of C to elts of nodes(TOPO)
%    [node  dir] where
%             node is index of elt in nodes(TOPO)
%             dir is 1 if data enters here, -1 if data leaves here

numnodes = length(TOPO.links) * 2;

C = zeros(numnodes,numnodes);

curnode = 0;
portmap = [];
for i = 1:length(TOPO.links)
        link = TOPO.links(i);
        curnode = curnode + 1;
        portmap(curnode,:) = [link.src -1];
        curnode = curnode + 1;
        portmap(curnode,:) = [link.dst 1];

        C(curnode-1,curnode) = link.bw;
end

c_node = 1;
c_dir = 2;

for i = 1:length(TOPO.nodes)
        ins = find(portmap(:,c_node) == i & portmap(:,c_dir) == 1);
        outs = find(portmap(:,c_node) == i & portmap(:,c_dir) == -1);

        for j = ins
                for k = outs
                        C(j,k) = inf;
                end
        end
end
function [a, b, c] = debug(t)

update(t);

fieldnames(t)

a = t.nodes
b = t.locs
c = t.links
function display(TOPO)
% DISPLAY a topo object
% a link is a unidirectional, so the value is probably twice what you want

fprintf('[TOPO object: %d nodes %d links]\n',...
        length(TOPO.nodes),length(TOPO.links));
function draw(TOPO)
% draw(topo)
%
% draw the topology figure in a new window

TOPO.cur_fig = figure;
axis(TOPO.axis);
axis equal;
axis manual;
box on;
hold on;

for i = 1:length(TOPO.nodes)
        nm = plot(TOPO.nodes{i}.loc(1),TOPO.nodes{i}.loc(2),'ob');
        TOPO.nodes{i}.mark_handle = nm;
        if (isfield(TOPO.nodes{i},'nameloc'))
                TOPO.nodes{i}.nameloc(3) = text(TOPO.nodes{i}.nameloc(1),...
                                TOPO.nodes{i}.nameloc(2),TOPO.nodes{i}.name);
        end
end

% yes, this draws the same link twice.  fix it if it matters -dam 11/21

TOPO.linkarray = [];
for i = 1:length(TOPO.links)
        TOPO.links(i).handle = drawlink(TOPO,TOPO.links(i));
        TOPO.linkarray = [TOPO.linkarray ; ...
                [ TOPO.links(i).src TOPO.links(i).dst TOPO.links(i).bw]];
end

assignin('caller',inputname(1),TOPO);
function ex(t)

t.nodes
function labelnames(TOPO)
% function labelnames(TOPO)
% make it easy to label the nodes

for i = 1:length(TOPO.nodes)
        fprintf('Place label for node %d "%s"\n',i,char(TOPO.nodes{i}.name));
```

```
		origcolor = get(TOPO.nodes{i}.mark_handle,'color');
		set(TOPO.nodes{i}.mark_handle,'color',[1 0 0]);

		if (isfield(TOPO.nodes{i},'nameloc'))
			good_x = TOPO.nodes{i}.nameloc(1);
			good_y = TOPO.nodes{i}.nameloc(2);
		end
		th = [];
		while (1)
			fprintf('Button 1 to (re)place text, Button 3 to accept\n');
			[x,y,button] = ginput(1);
			if (3 == button) break; end
			if (~isempty(th)) delete(th); end
			th = text(x,y,TOPO.nodes{i}.name);
			good_x = x; good_y = y;
		end
		TOPO.nodes{i}.nameloc = [good_x, good_y, th];
		set(TOPO.nodes{i}.mark_handle,'color',origcolor);
	end

	assignin('caller',inputname(1),TOPO);function names(TOPO)
	% NAMES the list of names of the nodes in the topo

	fprintf('Node\t\tName\n');
	for i = 1:size(TOPO.names,1)
		fprintf('%d\t\t%s\n',i,TOPO.names{i});
	end
	function [node] = nodes(TOPO)
	% function [node] = nodes(TOPO)
	%   returns a cell array describing nodes in the TOPO

	node = TOPO.nodes;
	function [TOPO] = topo(TOPO)
	% [TOPO] = topo(TOPO)
	%% if input TOPO is 'init', create a new topology
	%
	%          newtopo = topo('init');
	%
	%  else add new nodes to TOPO
	%
	%  nodes is a array of structs, one per node
	%  link is a array of structs, one per link
	%       a link is a unidirectional item, so there are probably twice
	%       as many links as you'd expect.
```

```
if (nargin < 1)
        error('topo(TOPO) or topo("init") - not enough args');
end

if (ischar(TOPO) & TOPO == 'init')
        clear TOPO

        TOPO.nodes = [];
        TOPO.links = [];

        TOPO.capacity = [];    % now computed as needed
        TOPO.locs = [];        % internal cache
        TOPO.linkarray = [];   % internal cache

        f = figure;
        axis([0 75 0 50 ]);
        TOPO.axis = axis;
        TOPO.cur_fig = f;
        axis equal
        axis manual
        box on
        hold
else
        figure(TOPO.cur_fig);
end



nodecount = length(TOPO.nodes);

while (1)
        clear nodeinfo;
        fprintf(1,'\n\nHit Button 3 to stop\n\n');
        [x y but] = ginput(1);
        if (but == 3) break; end
        x = floor(x); y = floor(y);
        nm = plot(x,y,'ob');
        name = input('Enter name > ','s');

        nodeinfo.loc = [ x y];
        nodeinfo.mark_handle = nm;
        nodeinfo.name = cellstr(name);
        nodecount = nodecount + 1;
        TOPO.nodes{nodecount} = nodeinfo;
end
```

```
if ('topo' ~= class(TOPO))
        TOPO = class(TOPO,'topo');
end

if (nargout == 0)
        assignin('caller',inputname(1),TOPO);
end
function lh = drawlink(TOPO, link)
% assumes TOPO.linkarray is already valid, and draws the position of
%  link line based on the number of links already present in linkarray

c_src = 1;
c_dst = 2;
c_bw = 3;

i = link.src;
j = link.dst;

x1 = TOPO.nodes{i}.loc(1);
y1 = TOPO.nodes{i}.loc(2);
x2 = TOPO.nodes{j}.loc(1);
y2 = TOPO.nodes{j}.loc(2);


if (isempty(TOPO.linkarray))
        num_links = 0;
else
        num_links = sum(TOPO.linkarray(:,c_src) == i & TOPO.linkarray(:,c_dst) == j);
end

pattern = [ 0 1 -1 2 -2 3 -3] * .3;

if (abs(x1 - x2) > abs(y1 - y2))
        delta_x = 0;
        delta_y = pattern(num_links + 1);
else
        delta_x = pattern(num_links + 1);
        delta_y = 0;
end

lh = line([x1 x2] + delta_x, [y1 y2] + delta_y, 'color', 'black');
function update(TOPO)
```

```
clear TOPO.locs;
for i = 1:length(TOPO.nodes)
        TOPO.locs(i,:) = TOPO.nodes{i}.loc
end

clear TOPO.linkarray;
for i = 1:length(TOPO.links)
        TOPO.linkarray = [TOPO.linkarray ; ...
                [ TOPO.links(i).src TOPO.links(i).dst TOPO.links(i).bw]];
end

% these are here to be cut and pasted into other functions as needed
%  there doesn't seem to be a good way to pass them around in another fashion
%  (using assinging('caller'...) to force their definition sounds like asking
%   for trouble 'cause you'll overwrite another definition of them...)
c_src = 1;
c_dst = 2;
c_bw = 3;

assignin('caller',inputname(1),TOPO);
```